**SOM 2.0**

Dockers

Development Setup

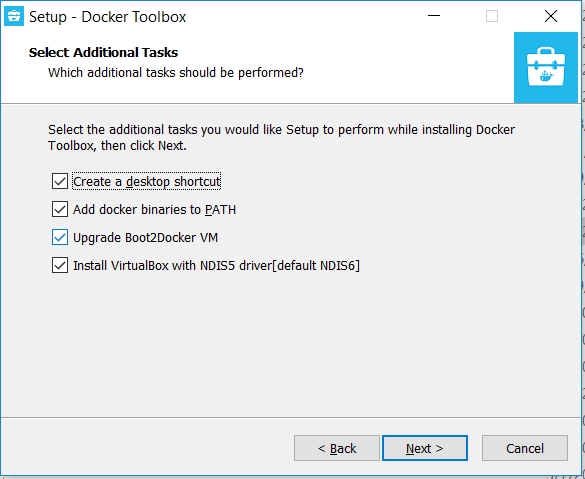
And

Environment

**Dockers Environment Requirement:**

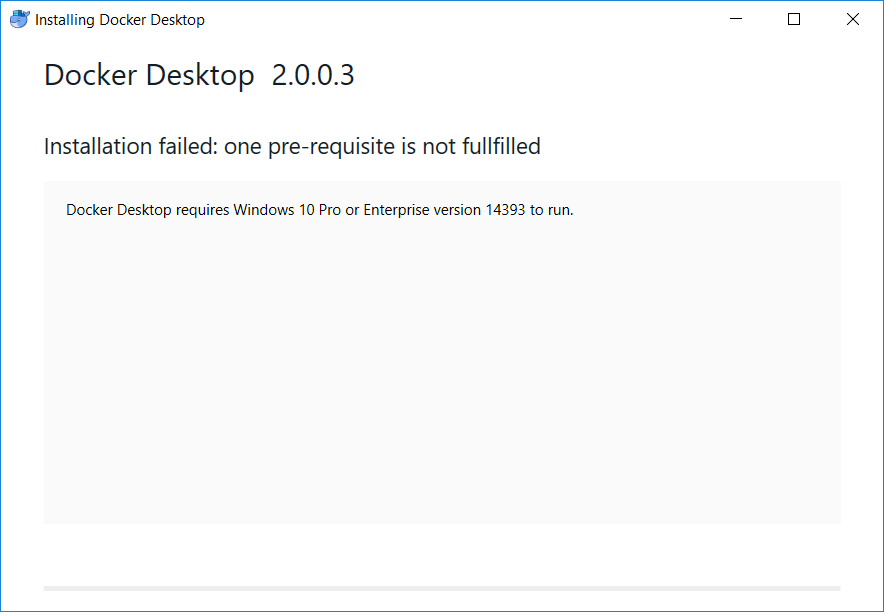
**Software’s Required:**

**Windows 10 Home/XP/Win 7 – Docker ToolBox.exe – Install Virtual Box with NDIS5 driver**

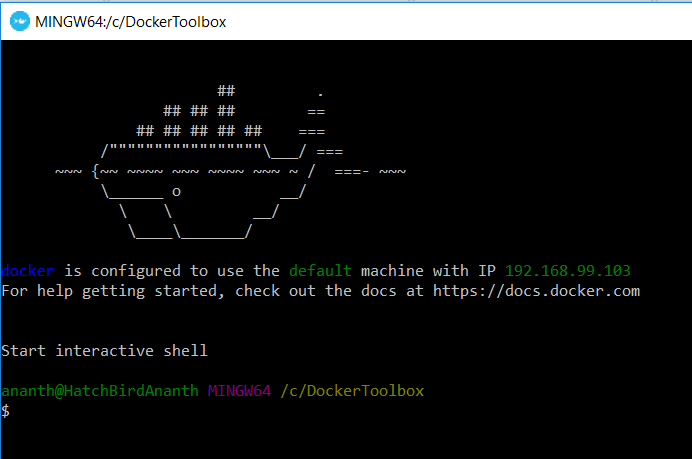
** **

Website: <https://docs.docker.com/docker-for-windows/install/>

**Windows Pro / Enterprise : Docker for Windows Installer.exe**

****

**On Windows Installation execute Docker Quickstart Terminal**

****

**For Dockers Cassandra: (Currently Not Applicable, if required we can set database in docker)**

**Note: Cassandra occupies 1.5 GB Memory in Normal Environment.**

**As Daemon Service: -d parameter**

$ docker run --name cassandra -d -e CASSANDRA\_BROADCAST\_ADDRESS=192.168.99.100 -p 9042:9042 cassandra:latest

$ docker run -it --link cassandra:cassandra --rm cassandra sh -c 'exec cqlsh "$CASSANDRA\_PORT\_9042\_TCP\_ADDR"'

$ docker run -it --link cassandra:cassandra --rm cassandra sh -c 'exec cqlsh "$CASSANDRA\_BROADCAST\_ADDRESS"'

$ docker run --name cassandra -d -e CASSANDRA\_BROADCAST\_ADDRESS=<ip-address> -p 9042:9042 cassandra:latest

**OR**

**Local IP Environment for CQLSH:**

$ docker run -it --link cassandra:cassandra --rm cassandra sh -c 'exec cqlsh "192.168.99.100"'

**Static IP Environment for CQLSH:**

$ docker run -it --link cassandra:cassandra --rm cassandra sh -c 'exec cqlsh "52.22.83.229"'

**To Start ZOOKEEPER –** confluentinc/cp-zookeeper

**Note: Zookeeper/Kafka occupies 1.0 GB Memory in Normal Environment.**

$ docker run -p 2181:2181 -d --net=host --name=zookeeper -e ZOOKEEPER\_CLIENT\_PORT=2181 confluentinc/cp-zookeeper:4.0.0 **OR**

$ docker run -p 2181:2181 -d --rm zookeeper -e ZOOKEEPER\_CLIENT\_PORT=2181 confluentinc/cp-zookeeper:4.0.0 **OR**

$ docker startzookeeper

**To Start KAFKA -** confluentinc/cp-kafka:4.0.0

**Local IP Environment**

$ docker run -p 9092:9092 -d --net=host --name=kafka \

-e KAFKA\_ZOOKEEPER\_CONNECT=192.168.99.100:2181 \

-e KAFKA\_ADVERTISED\_LISTENERS=PLAINTEXT://192.168.99.100:9092 \

-e KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR=1 confluentinc/cp-kafka:4.0.0

**AWS Static IP Environment**

$ docker run -p 9092:9092 -d --net=host --name=kafka \

-e KAFKA\_ZOOKEEPER\_CONNECT=172.17.0.1:2181 \

-e KAFKA\_ADVERTISED\_LISTENERS=PLAINTEXT://172.17.0.1:9092 \

-e KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR=1 confluentinc/cp-kafka:4.0.0

**OR**

$ docker startkafka

**To Create Application Topics**

**Local IP Environment** for Queue Topics

$ docker run --net=host --rm confluentinc/cp-kafka:4.0.0 kafka-topics --create --topic internal-topic --partitions 2 --replication-factor 1 --if-not-exists --zookeeper 192.168.99.100:2181

$ docker run --net=host --rm confluentinc/cp-kafka:4.0.0 kafka-topics --create --topic message-topic --partitions 1 --replication-factor 1 --if-not-exists --zookeeper 192.168.99.100:2181

**Static IP Environment** for Queue Topics

$ docker run --net=host --rm confluentinc/cp-kafka:4.0.0 kafka-topics --create --topic internal-topic --partitions 2 --replication-factor 1 --if-not-exists --zookeeper 172.17.0.1:2181

$ docker run --net=host --rm confluentinc/cp-kafka:4.0.0 kafka-topics --create --topic message-topic --partitions 1 --replication-factor 1 --if-not-exists --zookeeper 172.17.0.1:2181

**Steps To Create Docker Images**

**Step1:** Check Docker image already exists in local using

**$ docker images –a** command.

**Step2:** If corresponding image exists clean the existing image using

**$ docker rmi –f <image ids… separate by spaces>**

**Step3:** Open Docker ToolBox terminal and change directory to respective build folder where **Dockerfile** resides.

**Step4:** Execute command **$ docker image build –t <image name> .**

**Step5:** Tag the created docker image using following command

**$ docker tag <image-name> <tag path>:<tag name>**

**Note: Clean images in Local, Docker Hub and AWS Cloud for updated new images using**

**$ docker rmi –f <image ids…> in local docker toolbox terminal and AWS putty terminal**

**For more Docker commands refer:** <https://docs.docker.com/docker-for-windows/install/>

1. **Docker Image Create on Local for Eureka Server**

$ docker image build -t som-eureka-server . (Period ‘.’ Is Mandatory)

$ docker tag som-eureka-server tagsomsivisoft/som2.0:som-eureka-server (For Local)

$ docker tag som-eureka-server tagsomsivisoft/som2.0.uat:som-eureka-server (For UAT)

**Note: Need a Docker Account in** [**https://cloud.docker.com/repository/list**](https://cloud.docker.com/repository/list)**.**

**Login with Docker Hub Credentials. For example tagsomsivisoft**

$ docker login

$ docker push tagsomsivisoft/som2.0:som-eureka-server (For Repository 1 to deploy in AWS)

$ docker push tagsomsivisoft/som2.uat:som-eureka-server (For Repository 2 to deploy in AWS)

**To run in Local DockerToolBox.exe**

$ docker run -p 8761:8761 -d som-eureka-server

OR

**To run in AWS environment (in Putty) for Log view**

$ docker run -p 8761:8761 tagsomsivisoft/som2.0:som-eureka-server

OR

**To run in AWS environment (in Putty) as daemon service**

$ docker run -p 8761:8761 -d tagsomsivisoft/som2.0:som-eureka-server

$ docker run --rm -e JAVA\_OPTS='-Xmx128m' -p 8761:8761 -d tagsomsivisoft/som2.0:som-eureka-server

1. **Docker Image Create on Local for OAuth Server**

$ docker image build -t som-oauth-server . (Period ‘.’ Is Mandatory)

$ docker tag som-oauth-server tagsomsivisoft/som2.0:som-oauth-server (For Local)

$ docker tag som-oauth-server tagsomsivisoft/som2.0.uat:som-oauth-server (For UAT)

**Note: Need a Docker Account in** [**https://cloud.docker.com/repository/list**](https://cloud.docker.com/repository/list)**.**

**Login with Docker Hub Credentials. For example tagsomsivisoft**

$ docker login

$ docker push tagsomsivisoft/som2.0:som-oauth-server (For Repository 1 to deploy in AWS)

$ docker push tagsomsivisoft/som2.uat:som-oauth-server (For Repository 2 to deploy in AWS)

**To run in Local DockerToolBox.exe**

$ docker run -p 8090:8090 -d som-oauth-server

OR

**To run in AWS environment (in Putty) for Log view**

$ docker run -p 8090:8090 tagsomsivisoft/som2.0:som-oauth-server

OR

**To run in AWS environment (in Putty) as daemon service**

$ docker run -p 8090:8090 -d tagsomsivisoft/som2.0:som-oauth-server

$ docker run --rm -e JAVA\_OPTS='-Xmx128m' -p 8090:8090 -d tagsomsivisoft/som2.0:som-oauth-server

1. **Docker Image Create on Local for SOM-Admin**

$ docker image build -t som-admin . (Period ‘.’ Is Mandatory)

$ docker tag som-admin tagsomsivisoft/som2.0:som-admin (For Local)

$ docker tag som-admin tagsomsivisoft/som2.0.uat:som-admin (For UAT)

**Note: Need a Docker Account in** [**https://cloud.docker.com/repository/list**](https://cloud.docker.com/repository/list)**.**

**Login with Docker Hub Credentials. For example tagsomsivisoft**

$ docker login

$ docker push tagsomsivisoft/som2.0:som-admin (For Repository 1 to deploy in AWS)

$ docker push tagsomsivisoft/som2.uat:som-admin (For Repository 2 to deploy in AWS)

**To run in Local DockerToolBox.exe**

$ docker run -p 8100:8100 -d som-admin

OR

**To run in AWS environment (in Putty) as daemon service**

$ docker run --rm -e JAVA\_OPTS='-Xmx256m' \

-e somEnvironment='uat' \

-e somDomainUrl='http://ec2-52-22-83-229.compute-1.amazonaws.com' \

-e somUiServerUrl='http://ec2-52-22-83-229.compute-1.amazonaws.com' \

-e somEurekaPort='8761' \

-e somEurekaContext='som-eureka-server' \

-e somAdminPort='8100' \

-e somAdminContext='som-admin-service' \

-e somSenderPort='8200' \

-e somSenderContext='som-sender-service' \

-e LOG4J.DIR='/home/ec2-user/logs' \

-p 8100:8100 -d tagsomsivisoft/som2.0:som-admin

1. **Docker Image Create on Local for SOM-Sender**

$ docker image build -t som-sender . (Period ‘.’ Is Mandatory)

$ docker tag som-sender tagsomsivisoft/som2.0:som-sender (For Local)

$ docker tag som-sender tagsomsivisoft/som2.0.uat:som-sender (For UAT)

**Note: Need a Docker Account in** [**https://cloud.docker.com/repository/list**](https://cloud.docker.com/repository/list)**.**

**Login with Docker Hub Credentials. For example tagsomsivisoft**

$ docker login

$ docker push tagsomsivisoft/som2.0:som-sender (For Repository 1 to deploy in AWS)

$ docker push tagsomsivisoft/som2.uat:som-sender (For Repository 2 to deploy in AWS)

**To run in Local DockerToolBox.exe**

$ docker run -p 8200:8200 -d som-sender

OR

**To run in AWS environment (in Putty) for Log view**

$ docker run -p 8200:8200 tagsomsivisoft/som2.0:som-sender

OR

**To run in AWS environment (in Putty) as daemon service**

$ docker run -p 8200:8200 -d tagsomsivisoft/som2.0:som-sender

$ docker run --rm -e JAVA\_OPTS='-Xmx256m' -p 8200:8200 -d tagsomsivisoft/som2.0:som-sender

1. **Docker Image Create on Local for SOM-View**

$ docker image build -t som-view . (Period ‘.’ Is Mandatory)

$ docker tag som-view tagsomsivisoft/som2.0:som-view (For Local)

$ docker tag som-view tagsomsivisoft/som2.0.uat:som-view (For UAT)

**Note: Need a Docker Account in** [**https://cloud.docker.com/repository/list**](https://cloud.docker.com/repository/list)**.**

**Login with Docker Hub Credentials. For example tagsomsivisoft**

$ docker login

$ docker push tagsomsivisoft/som2.0:som-view (For Repository 1 to deploy in AWS)

$ docker push tagsomsivisoft/som2.uat:som-view (For Repository 2 to deploy in AWS)

**To run in Local DockerToolBox.exe**

$ docker run -p 8300:8300 -d som-view

OR

**To run in AWS environment (in Putty) for Log view**

$ docker run -p 8300:8300 tagsomsivisoft/som2.0:som-view

OR

**To run in AWS environment (in Putty) as daemon service**

$ docker run -p 8300:8300 -d tagsomsivisoft/som2.0:som-view

$ docker run --rm -e JAVA\_OPTS='-Xmx256m' -p 8300:8300 -d tagsomsivisoft/som2.0:som-view

1. **Docker Image Create on Local for SOM-UI ( Visual Code Angular JS Application)**

$ docker image build -t somui . (Period ‘.’ Is Mandatory)

$ docker tag somui tagsomsivisoft/som2.0:somui (For Local)

$ docker tag somui tagsomsivisoft/som2.0.uat:somui (For UAT)

**Note: Need a Docker Account in** [**https://cloud.docker.com/repository/list**](https://cloud.docker.com/repository/list)**.**

**Login with Docker Hub Credentials. For example tagsomsivisoft**

$ docker login

$ docker push tagsomsivisoft/som2.0:somui (For Repository 1 to deploy in AWS)

$ docker push tagsomsivisoft/som2.0.uat:somui (For Repository 2 to deploy in AWS)

**To run in Local DockerToolBox.exe**

$ docker run -p 80:80 -d somui

OR

**To run in AWS environment (in Putty) for Log view**

$ docker run -p 80:80 tagsomsivisoft/som2.0:somui

OR

**To run in AWS environment (in Putty) as daemon service**

$ docker run -p 80:80 -d tagsomsivisoft/som2.0:somui

$ docker run -p 80:80 -d tagsomsivisoft/som2.0.uat:somui

**Find outs, Tips and Resolutions**

Error response from daemon: Get https://registry-1.docker.io/v2/: dial tcp: lookup registry-1.docker.io on 10.0.2.3:53: read udp 10.0.2.15:57002->10.0.2.3:53: i/o timeout

**Docker ToolBox Issue**

**$** **docker-machine restart**

**Shell Scripts for starting SOM2.0 Application in AWS Cloud.**



**# For Connecting Docker Container**

$ docker ps (To get image id)

$ docker stop <image-id1 image-id2 …> (To Stop docker containers)

$ docker exec -it <image-id> /bin/bash

**# Delete all containers**

$ docker rm $(docker ps -a -q)

$ docker ps -a | grep som | awk '{print $1}'

$ docker rm $(docker ps -a | grep som | awk '{print $1}')

**# Delete all images**

$ docker rmi $(docker images -q)